Jukka Peltola

EEM2-siltasovelluksen päivitys uudelle alustalle

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Auto- ja kuljetustekniikka

Insinöörityö

5.5.2013

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Jukka Peltola EEM2-siltasovelluksen päivitys uudelle alustalle 36 sivua + 13 liitettä 5.5.2012
Tutkinto	insinööri (AMK)
Koulutusohjelma	Auto- ja kuljetustekniikan koulutusohjelma
Suuntautumisvaihtoehto	Autosähkötekniikka
Ohjaaja(t)	Ari Konttinen, Development Manager, AGCO Power Vesa Linja-aho, autoelektroniikan lehtori

Insinöörityössä kartoitettiin uusia alustaratkaisuja AGCO Powerin moottorintestausjärjestelmän siltasovellukselle. Työn tavoitteena oli löytää nykyisen siltasovelluksen alustalle sopiva korvaaja ja siirtää siltasovellus siihen.

Siltasovellus on ohjelma, jonka päätehtävä on luoda rajapinta eri kommunikointiprotokollia noudattavien moottorintestausjärjestelmän osapuolten välille ja täten mahdollistaa osapuolten välinen tiedonsiirto. Käytännössä tämä tarkoittaa, että siltasovellus yhdistää ja muuntaa osajärjestelmiin hajautuneen tiedon kaikille järjestelmän osapuolille sopivaan muotoon.

Tutkittavien alustojen lukumäärä rajattiin kolmeen ratkaisuvaihtoehtoon. Valittujen alustojen sopivuutta siltasovelluksen uudeksi alustaksi tutkittiin ensin alustakohtaisesti ja tämän jälkeen niitä vertailtiin keskenään. Vertailuissa kiinnitettiin huomiota erityisesti ratkaisun toimintavarmuuteen, käyttöikään ja päivitettävyyteen. Ratkaisua valittaessa otettiin huomioon myös käytettävissä olevat resurssit ja ratkaisun yksinkertaisuus. Vertailun parhaalle alustalle laadittiin uuden siltasovelluksen prototyyppi.

Työssä suunniteltiin ja toteutettiin uuden välimoduulin prototyyppi PCAN-MicroMod Mix 3 -moduuliyksikköön. Siltasovellus suunniteltiin C-ohjelmointikielellä, käyttäen apuna nykyisen siltasovelluksen lähdekoodia. Prototyypin siltasovellusta testattiin kehitysohjelmiston testaustyökaluilla ja PCAN-MicroMod-sarjan kehitysalustalla.

Avainsanat	moottorintestausjärjestelmä, siltasovellus, välimoduuli

Author(s) Title Number of Pages Date	Jukka Peltola Porting EEM2 Bridge Application to a New Platform 36 pages + 13 appendices 5 May 2013
Degree	Bachelor of Engineering
Degree Programme	Automotive and Transport Engineering
Specialisation option	Automotive Electronics Engineering
Instructor(s)	Ari Konttinen, Development Manager, AGCO Power Vesa Linja-aho, Senior Lecturer in Automotive Electronics

This Bachelor's thesis was carried out in order to discover new platform possibilities for a 'Bridge application' used in the AGCO Power's engine testing system. The objective was to find a substitutive platform for the existing, soon to be outdated platform, and port the Bridge application into it.

The Bridge application is a program running in a 'Bridge' module. Its primary function is to create an interface between the engine testing system's subsystems, and thus allow communication between subsystems operating on different communication protocols. Practically this means that the Bridge module combines the information that is scattered to subsystems and converts it into a form that is interpretable by the information user.

The number of platforms to be researched was limited to three solutions. The suitability of the selected platforms was first examined individually and after that all solutions were compared among each other. In the examination, particular attention was paid in to solutions reliability, service life, updatability and simplicity. The best solution of comparison was taken for further development

The new Bridge application's prototype was developed in a PCAN-MicroMod Mix 3 module. The application was programmed using the C-programming language. The current Bridge application's source code was used as a base for the new application. The prototype was tested using software testing tools and PCAN-MicroMod development board.

Keywords	engine testing system, gateway module, bridge application

Sisällys

Lyhenteet, termit ja määritelmät

EDC	Electronic Diesel Control
ECU	Electronic Control Unit
EEM	Electronic Engine Management
CAN	Controller Area Network
HSCAN	High Speed CAN
SAE-J1939	Standardikokoelma J1939
PGN	Parameter Group Number
SPN	Suspect Parameter Number
TSC1	Torque Speed Control 1 (SAE-J1939 PGN0)
WinEEM	Moottorinohjausjärjestelmän kehitystyökalu EEM
USB	Universal Serial Bus

1 Johdanto

Tämä insinöörityö tehtiin AGCO Powerille. Työn tarkoituksena oli kartoittaa uusia alustaratkaisuja AGCO Powerin moottorintestausjärjestelmän siltasovellukselle ja toteuttaa yhdelle tutkituista alustaratkaisuista uuden siltasovelluksen prototyyppi.

Siltasovellus on ohjelma, joka pyörii *välimoduuliksi* kutsutulla alustalla (ts. laitteistolla). Sen päätehtävänä on mahdollistaa eri kommunikointiprotokollia noudattavien moottorintestausjärjestelmän osapuolten välinen tiedonsiirto. Välimoduulia voidaan pitää osapuolten välisenä ”tulkkina”, joka yhdistää ja muuntaa osajärjestelmiin hajautuneen tiedon kaikille järjestelmän osapuolille sopivaan muotoon. Vastaavanlaiset moduulit ovat yleisiä erityisesti hajasijoitetun tiedon sovelluksissa, joissa dataintegraatio on toteutettava jossakin keskitetyssä pisteessä. Niitä löytyy erityisesti autoteollisuuden sovelluksista (ajoneuvoissa nk. yhdyskäytävämoduulit).

Nykyinen siltasovellus on toteutettu AGCO Powerin toisen sukupolven EEM2-moottorinohjausjärjestelmän moottorinohjausyksikköön, josta alkuperäinen moottorinohjausohjelma on korvattu C-ohjelmointikielellä laaditulla ohjelmalla. EEM2 -moottorinohjausyksikön tuotanto päättyi vuonna 2005. Jäljellä olevien EEM2-yksiköiden vähäisen määrän ja iän takia siltasovellukselle tarvittiin uusi alusta.

Insinöörityössä käsitellään ensin lyhyesti dieselmoottorin ohjausjärjestelmän ja CAN-väylän toimintaperiaatteita. Tämän jälkeen tutustutaan AGCO Powerin käytössä olevaan moottorintestausjärjestelmään ja syvennytään nykyisen välimoduulin toimintaperiaatteisiin testausjärjestelmän osana. Seuraavaksi kartoitetaan uusia alustaratkaisuja siltasovellukselle. Valittujen alustaratkaisujen välillä suoritetaan vertailua ja lopuksi yhdelle alustalle laaditaan välimoduulin toteuttava prototyyppi.

1.1 Työkulku

Työ aloitettiin tutustumalla käytössä olevaan välimoduuliin AGCO Powerin tuotekehitysyksikössä. Tarkoituksena oli kerätä mahdollisimman paljon tietoa sekä nykyisestä järjestelmästä että tulevaan järjestelmään kohdistuvista vaatimuksista ja toiveista. Nykyistä välimoduulia testattiin tuotekehitysyksikön sähkölaboratorion

laitteiston avulla ja sen todellisessa toimintaympäristössä, kokoonpanohallin moottorintestausjärjestelmän osana.

Seuraava vaihe oli etsiä uusia mahdollisia ratkaisuvaihtoehtoja siltasovelluksen uudeksi alustaksi. Ratkaisuvaihtoehtojen määrä rajattiin kolmeen mahdolliseen, uuden välimoduulin toteuttavaan ratkaisuvaihtoehtoon. Vaihtoehtojen välillä suoritettiin vertailua, minkä jälkeen vertailun parhaalle alustalle suunniteltiin ja toteutettiin uuden välimoduulin prototyyppi.

1.2 AGCO Power

AGCO on kansainvälinen maatalouslaitteiden suunnitteluun, valmistukseen ja jakeluun erikoistunut konserni, jolla on toimintaa yli sadassa eri maassa. Suomessa toimii konsernin dieselmoottoreiden suunnitteluun ja valmistukseen erikoistunut yritys AGCO Power. Yrityksen tuotekehitys on keskittynyt Suomen toimipisteen, Nokialla sijaitsevan tehdasalueen yhteyteen. Yrityksellä on myös tuotantolaitokset Brasiliassa ja Kiinassa.

Nokian Linnavuoressa sijaitseva AGCO Powerin tehdasalue on toiminut 70 vuotta nykyisellä toimipaikallaan. Tehdasalueen omistus on kuitenkin vaihtunut vuosien varrella useaan otteeseen. Alun perin tehdas on toiminut Valmetin ja Sisun tehtaana ja rakennettu sotien aikana tuottamaan sotakalustoa, erityisesti lentokoneiden ja moottoriajoneuvojen moottoreita ja kulutusosia.

Nykypäivän AGCO Power valmistaa vuodessa noin 30 000 dieselmoottoria. Moottorit menevät usealle tunnetulle traktoreiden ja maatalouskoneiden valmistajalle mm. Valtralle ja John Deerelle. Nykyinen moottorivalikoima koostuu 3-, 4-, 6- ja 7-sylinterisistä moottoreista, jotka tarjoavat 50–500 hevosvoiman tehonlähteen 3,3–8,0 litran iskutilavuudella.

2 Teoria

2.1 Elektronisesti ohjattu dieseljärjestelmä

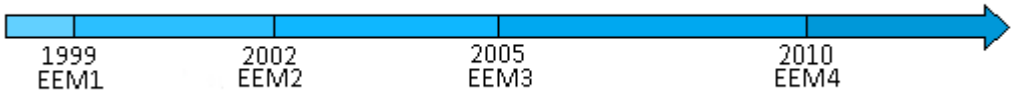
Dieselmoottoreiden nopeasti tiukentuvat typenoksidien ja hiukkaspäästöjen päästörajoitukset sekä valmistajien välinen kilpailu suorituskykyisimmästä, taloudellisimmasta ja vähäpäästöisimmästä moottorista asettavat korkeat tavoitteet moottorinohjausjärjestelmän kehitykselle. Modernissa dieselmoottorissa edellä mainittuihin tavoitteisiin päästään ainoastaan elektronisesti ohjatulla dieseljärjestelmällä (*lyh. EDC, Electronic Diesel Control*). EDC-järjestelmän päätehtävänä on kontrolloida polttonesteen ruiskutusmäärää, ruiskutushetkeä sekä suihkutuspainetta. [1, s. 100.]

Moottorinohjausjärjestelmä voidaan jakaa kolmeen osajärjestelmään: tunnistimet, toimilaitteet ja moottorinohjausyksikkö (kuva 1). Moottorinohjausyksikkö kerää tietoa moottorin tunnistimilta, toimilaitteilta sekä muilta ajoneuvon ohjausyksiköiltä ajoneuvoväylien välityksellä. Osajärjestelmiltä kerätyn tiedon ja moottorinohjausyksikön muistiin tallennettujen kartastojen ja algoritmien perusteella moottorinohjausyksikkö kykenee määritelemään moottorin hetkellisen toimintatilan ja mahdollisten säätöjen vaatimat toimenpiteet. Elektronisen moottorinohjausjärjestelmän sulauttaminen muiden ajoneuvon osajärjestelmien kanssa onkin edellytys tehokkaan toiminnan saavuttamiseksi. [1, s. 101.]

Kuva 1. Bosch EDC-17 -moottorinohjausyksikkö

2.2 Electronic Engine Management (EEM)

AGCO Power käyttää tuottamiensa moottoreiden moottorinohjausjärjestelmästä nimitystä *Electronic Engine Management* (lyhyemmin *EEM*). Nykyinen, yrityksen neljännen sukupolven moottoreissa käytössä oleva EEM4-moottorinohjausjärjestelmä ja sitä edeltävä EEM3 on suunniteltu Boschin valmistaman, dieselmoottoreille suunnatun EDC17-moottorinohjausjärjestelmän ohelle. EEM3-järjestelmään siirryttiin vuonna 2005, joka oli ensimmäinen EDC-sarjaan pohjautuva moottorinohjausjärjestelmä (kuva 2). EDC-sarjan moottorinohjausjärjestelmiin siirryttiin pääasiassa siksi, että täysin oman moottorinohjausjärjestelmän kehitys laitteistosta ohjelmistoon olisi vaatinut suhteettoman paljon resursseja.



Kuva 2. EEM-moottorinohjausjärjestelmän kehitys

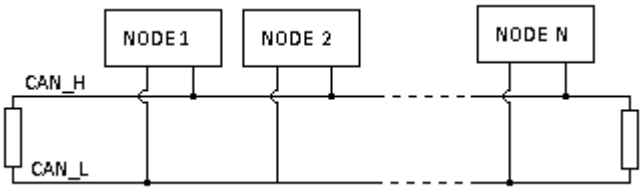
EEM2 ja sitä vanhemmat moottorinohjainyksiköt ovat yrityksen omaa kehitystyötä. Vanhan sukupolven EEM2-moottorinohjausyksiköt on otettu uusiokäyttöön moottorintestausjärjestelmien siltasovellusten alustoina eli välimoduuliyksikköinä.

2.3 Controller Area Network

CAN-väylä (engl. *Controller Area Network*) on alun perin Robert Bosch Gmbh:n kehittämä väyläratkaisu ajoneuvon toimilaitteiden ja moottorinohjausyksikön väliseen kommunikointiin. Väylä levisi nopeasti myös muualle teollisuuteen ja on tänä päivänä yksi käytetyimmistä automaatioväylistä ajoneuvoissa, koneissa ja teollisuuden ohjausjärjestelmissä. [2.]

CAN-väylä on usean isännän sarjaliikenneväylä (kuva 3), jossa kaikki väylän osapuolet eli solmut (engl.*node*) ovat toisiinsa nähden samanarvoisia ja näkevät kaiken väyläliikenteen (*multi-cast -väylä*). Samanarvoisuus ja *multi-cast* -väylärakenne tarkoittavat, että kaikki väylällä liikkuva data on kaikkien solmujen käytettävissä ja väylän ollessa vapaana, mikä tahansa solmu voi yrittää tiedonsiirtoa. Tämä johtaa

siihen, että ainoastaan kahden väylän osapuolen välinen peer-to-peer viestintä perinteisellä CAN-protokollalla ei ole mahdollista. [3.]



Kuva 3. CAN-väylän topologia ja päätevastukset

Väylällä kulkee kerrallaan ainoastaan yksi viesti. Viestin lähettäjä toimii väylän varaajana (*isäntä*), jolloin muut osapuolet (*orjat*) kuuntelevat väylää. Mikäli kaksi väylän osapuolta pyrkii lähettämään viestinsä väylälle samanaikaisesti, pienemmän prioriteetin omaava viesti pääsee väylälle ensimmäisenä, jolloin suuremman prioriteetin omaavan viestin lähettäjä siirtyy takaisin väylän kuuntelijaksi. Tällaista väylähierarkiaa kuvataan fraasilla *message oriented network* eli väylälle pyrkivän viestin prioriteetti toimii väylän varaavana tekijänä. [3.]

2.3.1 CAN-standardi (ISO-11898)

CAN-protokollan spesifikaatio on standardisoitu kansainvälisen ISO -standardiorganisaation (*International Organization of Standardization)* toimesta. CAN on dokumentoitu ISO-11898 -standardiin ja jaoteltu neljään osioon. Ensimmäinen osio ISO-11898-1 määrittelee CAN-protokollan eli miten esimerkiksi eri CAN-kehykset määritellään. Toinen osio (ISO-11898-2) määrittelee nopean väylän (*High Speed CAN*) fyysisen tason, kolmas (ISO-11898-3) vikasietoisen väylän (*Fault Tolerant CAN*) fyysisen tason ja neljännessä osiossa (ISO-11898-4) määritetään deterministinen TTCAN (*time-triggered communication*) -kommunikointiprotokolla. [4.]

CAN-kehykset

CAN käyttää neljää eri tyyppistä viestikehystä väylän tahojen väliseen kommunikointiin sekä väylän toimintatilan määritykseen. Kehykset ja niiden sisältö on määritelty tarkemmin ISO-11898-1 -standardissa. [5, s. 10.]

Neljä tiedonvälityksessä käytettävissä olevaa kehystä ovat

- sanomakehys
- kyselykehys
- virhekehys
- viivekehys.

Sanomakehys

Tämän työn kannalta ainoa oleellinen kehys on sanomakehys (kuva 4). Sanomakehys käyttää ISO-11898-1 -standardiin määriteltyä tiedonsiirron arkkitehtuuria. Karkeistettuna kehys koostuu otsikosta (engl. *header*) ja sitä seuraavasta välitettävästä sanomasta. Otsikko sisältää viestin tunnisteen (engl. *identifier*), joka on pituudeltaan joko 29-bittinen tai 11-bittinen riippuen siitä, kumpi CAN 2.0 -standardissa määritelty protokolla on käytössä. CAN 2.0B on nykyään yleisemmin käytetty protokolla ja kattaa molemmat tunnistepituudet. CAN 2.0A määrittelee ainoastaan lyhyemmän 11-bittisen tunnisteen. Tunniste pitää sisällään mm. viestin prioriteetin, jonka perusteella väylän varaus on toteutettu. [5, s. 10] Sanomakehys sisältää myös tunnistebittejä ja erinäisiä virheen tunnistusmenetelmiä, joihin ei tässä työssä perehdytä syvällisemmin. [5, s. 13]

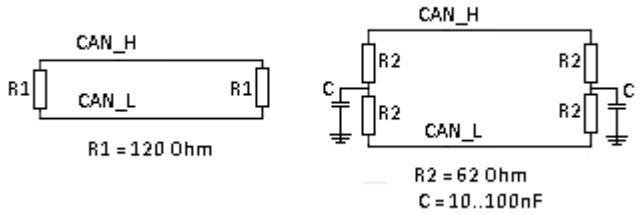
SOF	Start Of Frame	Kehyksen aloitusbitti
ID	Identifier	Viestin prioriteetin indikaattori
IDE	ID extension bit	Käytetty standardi (2.0A tai 2.0B)
DLC	Data Length code	Informaation määrä tavuina (0-8)
Data	-	Tieto-osio eli varsinainen informaatio
CRC	Cyclic redundancy check	Virheiden havaitsemismenetelmä
ACK S	Acknowledgement msg	Kuittausviesti
ACK D	End of Acknowledgement	Kuittauksen lopetus
EOF	End of Frame (stop bit)	Kehyksen lopetus

Kuva 4. CAN-standardin mukainen sanomakehys

2.3.2 Fyysinen rakenne (High Speed CAN)

Nopean CAN-väylän fyysinen taso on toteutettu kaksinapaisella parikaapelilla. Väylällä tulee käyttää fyysisen tason standardiin (ISO-11898-2) määriteltyä kaapeleiden kierrätystä: 40 kierrosta metriä kohden sekä päätevastuksia. Parikaapelille on määritelty myös maksimipituus, joka riippuu käytetystä tiedonsiirtonopeudesta. [4] Parikaapelin tarkoituksena on kumota kaapeleissa eri suuntiin kulkevista virroista syntyvät magneettivuot. Menetelmällä estetään johdinten välille syntyviä induktiivisen kytkeytymisen aiheuttamia häiriöitä. [6]

Väylän päätevastukset voidaan toteuttaa kahdella kuvan 5 osoittamalla tavalla. Kuvan vasemman puoleisessa piirissä väylän päihin on sijoitettu 120 ohmin vastukset. Kuvan oikealla puolella olevassa piirissä väylän päätevastuksena on kaksi 62 ohmin vastusta ja kondensaattori. Väylän päätevastuksilla estetään väylällä kulkevan signaalin heijastuminen väylän päästä. Heijastunut signaali aiheuttaa väylälle häiriötä ja on täten suodatettava väylältä mahdollisimman tehokkaasti. Kuvan oikean puolimmaisessa piirissä käytettävillä kondensaattoreilla suodatetaan väylällä esiintyviä korkeita häiriötaajuuksia. [4.]



Kuva 5. ISO 11898-2 -standardin mukaiset päätevastuskytkennät

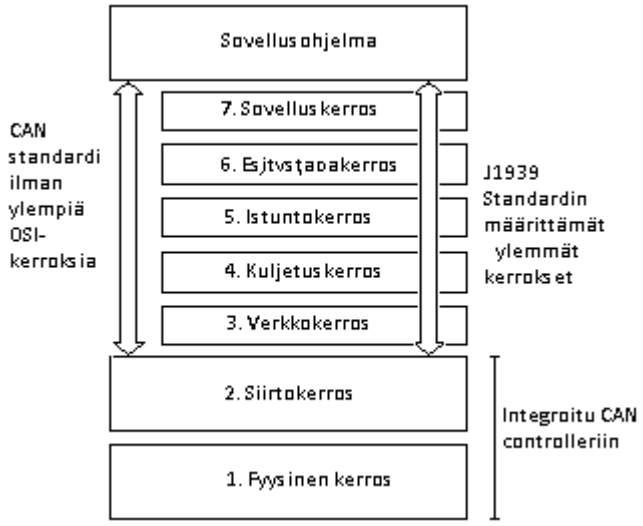
Väylän johtimista käytetään nimitystä CAN high (CAN_H) ja CAN low (CAN_L). Väylän käyttämät jännitetasot ovat standardikohtaisia, esimerkiksi High Speed CAN -väylässä lähetettäessä arvo 0, CAN high -johtimen jännite on 3,5 volttia ja CAN low -johtimen 1,5 volttia. Vastaavasti lähetettäessä 1, molempien johtimien jännite on 2,5 volttia (kuva 6). [3.]

Kuva 6. ISO 11898-2 -standardin mukaiset jännitetasot

2.3.3 Standardikokoelma SAE-J1939

Yhdysvaltalaisen *Society of Automotive Engineers* (SAE) -järjestön ylläpitämä standardikokoelma SAE-J1939 määrittelee, kuinka informaatiota välitetään CAN-väylässä toimilaitteiden ja moottorinohjausyksikön välillä. Se täydentää perinteistä, suhteellisen pinnallista CAN ISO-11898 -standardia erityisesti raskaan kaluston väylätarpeita varten. [7, s. 6.]

Kuvassa 7 esitetyllä OSI-mallilla havainnollistetaan tiedonsiirron eri kerroksia. Mallin jokainen kerros on yhteydessä ainoastaan yhtä ylempään ja yhtä alempaan kerrokseen. Jokaisella kerroksella on vastuu tarjota palveluita ainoastaan yhtä ylemmälle kerrokselle. Läheskään aina tiedonsiirrossa ei tarvita jokaista OSI-mallin kerrosta vaan hyödynnetään ainoastaan niitä kerroksia, joita tarvitaan. [8, s. 14.]



Kuva 7. OSI-malli

ISO-11898 -standardin mukainen CAN-protokolla määrittelee ainoastaan väylän fyysisen kerroksen sekä siirtokerroksen. Tämä on tarkoituksen mukaista, sillä erityisesti mikrokontrollerien rajallinen muistin määrä rajoittaa sovellukselle käytettävissä olevia resursseja, minkä takia protokolla on hyvä pitää mahdollisimman kompaktina. SAE J1939 -standardikokoelmassa määritetään nämä kuvassa 7 näkyvät viisi muuta tasoa, joilla tuodaan lisätoiminnallisuuksia tiedonsiirtoon. [8, s. 28.]

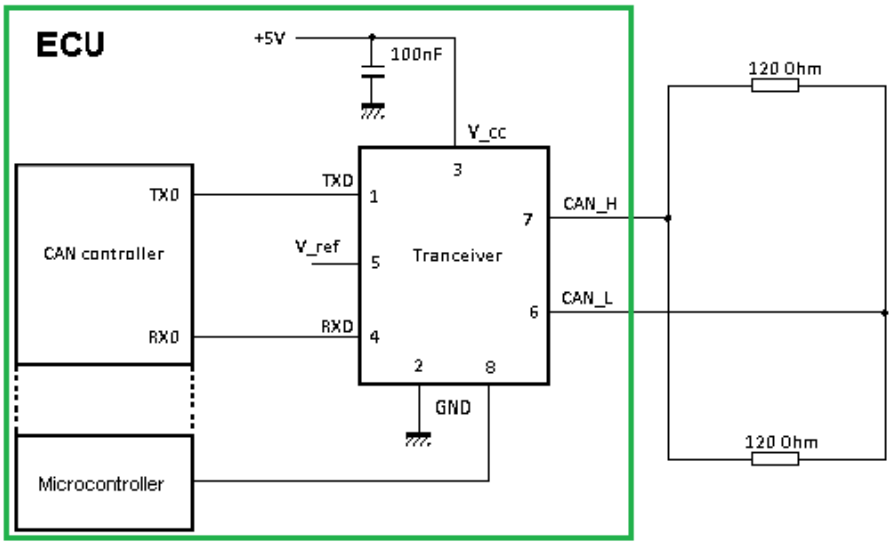
SAE J1939 -standardia noudattavassa sovelluksessa tietoa välitetään parametriryhmittäin. Parametriryhmän numero (PGN = Parameter Group Number) lisätään sanomakehyksen tunnisteosioon. [8, s. 26] Perinteisestä CAN-standardista poiketen J1939 -standardin 29-bittiseen tunnisteosioon on määritelty mm. sekä lähettäjän osoite että vastaanottajan osoite, jolloin väylän yli voidaan lähettää ainoastaan kahden osapuolen välisiä viestejä (*peer-to-peer* -viestitys). [8, s. 27] Yleensä jokainen parametriryhmä pitää sisällään joukon samankaltaisia parametreja. Parametrit on määritelty SAE J1939-71 -standardissa SPN (Suspect Parameter Number) -numeroin. Taulukossa 1 on esimerkkinä tässä työssä myöhemmin vastaan tuleva, SAE J1939 -standardia noudattava TSC1 (Torque Speed Control 1) -viesti. Kuten taulukosta 1 nähdään, viestissä välittyy yhteensä kymmenen eri parametria, joista jokainen liittyy jollakin tavalla moottorin vääntö- tai nopeuspyynnön välitykseen.

Taulukko 1. PGN0 Torque/Speed Control 1 (TSC1) [9, s. 947]

Start Position	Length	Parameter Name	SPN
1.1	2 bits	Engine Override Control Mode	695
1.3	2 bits	Engine Requested Speed Control Conditions	696
1.5	2 bits	Override Control Mode Priority	897
2-3	2 bytes	Engine Requested Speed/Speed Limit	898
4	1 byte	Engine Requested Torque/Torque Limit	518
5.1	3 bits	TSC1 Transmission Rate	3349
5.4	5 bits	TSC1 Control Purpose	3350
6.1	4 bits	Engine Requested Torque - High Resolution	4191
8.1	4 bits	Message Counter	4206
8.5	4 bits	Message Checksum	4207

2.3.4 Autoteollisuuden sovellukset

Tyypillinen CAN-väylään kytkettävä laite on elektroninen ohjausyksikkö (ECU). Kytketyn laitteen on kyettävä tulkitsemaan väylällä liikkuvia bittejä väylällä käytetyn standardin määrittämän viestirakenteen mukaisesti. Viestien parsiminen väylältä toteutetaan CAN controllerilla, jonka tehtävä on implementoida sovellukseen CAN-väylän fyysisen tason spesifikaatio viestirakenteen muodostamista varten. CAN controller voi olla joko integroituna mikrokontrolleriin tai erillinen IC-piiri (kuva 8).

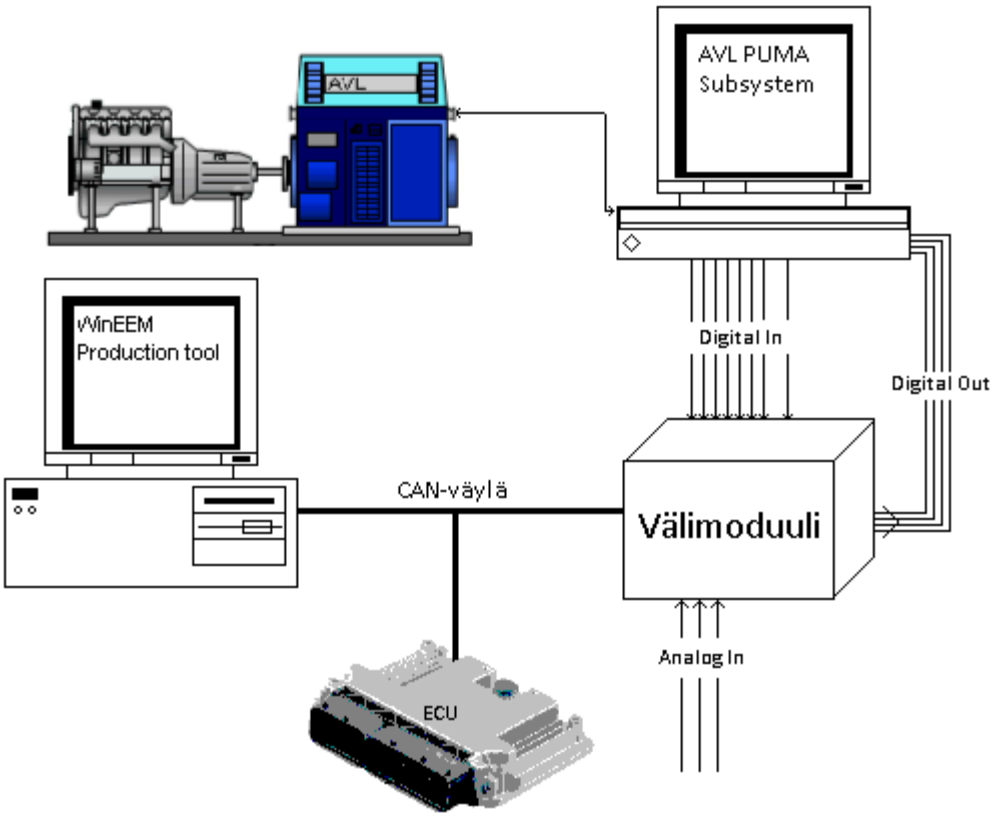


Kuva 8. CAN-väylän kytkentä sovellukseen

Yhteys CAN controllerin ja väyläjohdinten välille muodostetaan lähetinvastaanotinpiirillä (*engl. transceiver*). Kyseinen piiri pitää sisällään lähetys- ja vastaanottovahvistimen (*tranceiver = transmit and receive*) ja sen päätehtävä on muodostaa digitaalisesta informaatiosta sopivat jännitesignaalit väyläjohtimille. Lähetinvastaanotinpiiri sisältää yleensä myös mm. puskurin (*engl. buffer*), joka suojaa CAN controlleria ulkopuolisilta, korkeilta jännitepiikeiltä (EMI, ESD, transientit yms.).

3 Moottorintestausjärjestelmä

AGCO Powerin tuotannossa valmistuneiden moottoreiden toiminta testataan moottoreiden kokoonpanohallin testauspisteillä *AVL Test Bench* -moottorintestausjärjestelmällä (kuva 9). Testausprosessi aloitetaan asentamalla testattava moottori dynamometriin ja kytkemällä moottoriin moottorinohjausyksikkö sekä vaadittavat toimilaitteet. Testauksessa moottorin suorituskykyä testataan simuloimalla erilaisia toimintatilanteita dynamometrin vastamomenttia säätämällä. Menetelmällä tarkastetaan moottorin virheetön toiminta. Lisäksi testauksen tulosten pohjalta moottorikohtaiset, optimaaliset säätöparametrit tallennetaan moottorinohjausyksikköön, jolloin saavutetaan korkeampi moottorikohtainen suorituskyky.

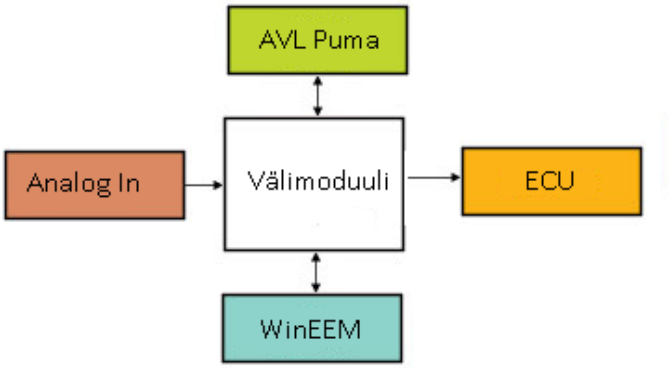


Kuva 9. Moottorintestausjärjestelmä

3.1 Järjestelmän komponentit

3.1.1 Välimoduuli

Välimoduulin tehtävä on muodostaa järjestelmän eri osajärjestelmien välille kommunikointiyhteys. Se toimii kuvainnollisesti osajärjestelmien välisenä ”tulkkina”, mahdollistaen eri kommunikointiprotokollia käyttävien osajärjestelmien välisen tiedonsiirron. Kuvassa 10 on esitetty välimoduulin välityksellä keskustelevat osajärjestelmät.



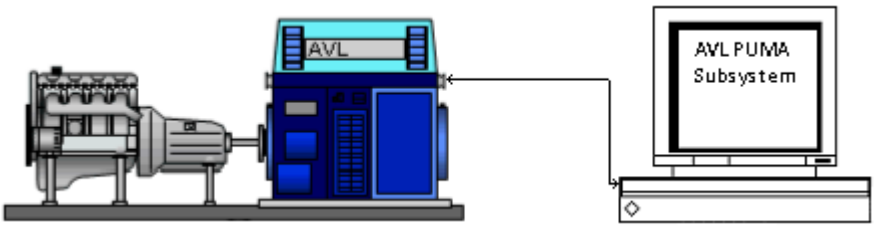
Kuva 10. Välimoduulin välityksellä keskustelevat komponentit

Nykyinen välimoduuli on toteutettu AGCO Powerin toisen sukupolven EEM2 -moottorinohjausjärjestelmän moottorinohjausyksikköön. Välimoduulille on räätälöity oma laiteohjelma (ts. siltasovellus), joka on ladattuna EEM2-moottorinohjausyksikön muistiin.

3.1.2 AVL PUMA

AVL on itävaltalaistaustainen, erityisesti dieselmoottoreiden testausjärjestelmien sekä instrumentointilaitteiden valmistukseen ja kehitykseen erikoistunut yritys. Yrityksen kehittämä moottoreiden testausjärjestelmä *AVL Test Bench* mahdollistaa polttomoottoreiden dynaamisen testauksen erilaisilla kuormituksilla, moottorikohtaisten optimaalisten säätöparametrien löytämiseksi. Testausjärjestelmä kerää tietoa moottorin toiminnasta sekä mahdollistaa kerätyn tiedon analysoinnin.

Moottorinestausjärjestelmä (kuva 11) koostuu moottoridynamometristä sekä AVL PUMA -ohjelmistosta. PUMAlla voidaan suorittaa dynamometrin ohjaustoimintoja, esimerkiksi valita haluttu ajosykli tai kytkeä mittakanavia käyttöön. Ohjelmistoon on toteutettu myös erillisiä komentoja WinEEM-ohjelmiston ohjausta varten. Näiden komentojen avulla PUMAlla voidaan pyytää WinEEMiä kytkemään esimerkiksi tietty vääntökartta käyttöön. PUMAn ja WinEEMin väliset ohjauskomennot ovat niin kutsuttuja PUMA-komentoja, jotka löytyvät listattuna liitteestä 1. PUMA-komennot välitetään välimoduulille rinnakkaisväylän välityksellä. Välimoduuli muodostaa PUMA-komennosta CAN-viestin ja välittää sen CAN-väylän yli WinEEMille.



Kuva 11. AVL-moottorintestausjärjestelmä

3.1.3 WinEEM4 Production tool

AGCO Powerilla käytetään moottorinohjausjärjestelmän diagnosointiin, monitorointiin, ohjelmistopäivitysten lataamiseen sekä testaus- ja kalibrointitoimintojen aktivointiin WinEEM-kehitysohjelmistoa. Moottorintestausjärjestelmässä WinEEM-ohjelmisto keskustelee moottorinohjausyksikön ja välimoduulin kanssa CAN-väylän ylitse.

3.2 Välimoduulin tehtävät

Välimoduulin päätehtävä on luoda kommunikointiyhteys moottorintestausjärjestelmän eri kommunikointiprotokollia noudattavien osajärjestelmien välille. Kommunikointiyhteys on toteutettu hyödyntäen moottorinohjausyksikön ja WinEEM-kehitysohjelmiston välistä CAN-väylää sekä PUMAn ja välimoduulin välille luotua rinnakkaisväylää. Välimoduulin tehtävät ovat seuraavat

- käsitellä analogista vääntöpyyntösignaalia ja välittää se (BR1-viestissä) WinEEMille sekä muodostaa vääntöpyynnöstä erillisen pyyntöviestin (BR5-viesti) perusteella SAE J1939 -standardin mukainen TSC1-viesti moottorinohjausyksikölle
- välittää PUMA-komennot WinEEMille (BR2-viesti).
- toteuttaa kättelytoimintoja järjestelmien välillä:
    - WinEEM - Välimoduuli
    - WinEEM - (Välimoduuli) - PUMA.

3.2.1 CAN-viestit

Välimoduulin ja WinEEMin välille on luotu seitsemän taulukossa 2 esitettyä CAN-viestiä, joilla järjestelmät välittävät tietoa toisilleen. Viesteistä BR1 on ainoa synkroninen, 50 millisekunnin välein väylälle lähetettävä viesti. Loput viesteistä ovat asynkronisia eli niiden lähettäminen on tilannekohtaista.

Taulukko 2. Järjestelmän CAN-viestit

Viestin nimi	**ID**	**Aikaväli [ms]**	**Lähde**	**Lisätiedot**
BR1	00FF5FFCh	50	Välimoduuli	Analoginen tulo
BR2	00FF5EFCh	asynkroninen	Välimoduuli	PUMA-komento, laskuri
BR3	0CFF5DF9h	asynkroninen	WinEEM	Kuittaus
BR4	0CFF5CF9h	asynkroninen	WinEEM	ECU versio- ja tilatiedot
BR5	0CFF5BF9h	asynkroninen	WinEEM	TSC1 käsittely
BR6	0CFF5AF9h	asynkroninen	WinEEM	Viestityksen aloituspyyntö
BR7	0CFF59Fh	asynkroninen	Välimoduuli	Vastaus aloituspyyntöön

BR1-viesti välittää maksimissaan kolmen analogisen signaalin arvot WinEEMille. Jokainen analoginen tulo muunnetaan välimoduulin AD-muuntimessa digitaaliseen muotoon ja skaalataan siltasovelluksessa 10 bitin resoluutiolle (0..1023).

Tällä hetkellä järjestelmän käytössä on ainoastaan yksi analoginen tulo, jolle on varattu BR1-viestistä (taulukko 3) kaksi ensimmäistä tavua (Analog In 1). Tämä signaali on joko nopeus- tai vääntöpyyntö väliltä 0–100% riippuen siitä kumpaa pyyntövaihtoehtoa moottorin ohjausjärjestelmässä käytetään. Analogiset tulot ovat käytännössä *ALV Test Bench* -testausjärjestelmän ohjauspaneelissa olevien potentiometrien antamat jännitesignaalit.

Taulukko 3. BR1-viesti

Tavu	Parametri
0,1	Analog In 1
2,3	Analog In 2
4,5	Analog In 3

BR2-viestissä (taulukko 4) välitetään PUMA-komento WinEEMille. Komento välitetään viestin ensimmäisessä tavussa. Viestin toisessa tavussa välitetään tieto viestilaskurin arvosta. Viestilaskuria kasvatetaan jokaisen lähetetyn BR2-viestin yhteydessä. Viestilaskurin arvon perusteella WinEEM ja välimoduuli kykenevät tulkitsemaan lähetettyjen ja vastaanotettujen viestien suhteen, ts. onko jokin viesti jäänyt saapumatta.

Taulukko 4. BR2-viesti

Tavu	Parametri
0	PUMA-komennon ID
1	Viestilaskuri

WinEEM lähettää BR3-viestin (taulukko 5) kuittauksena välimoduulin lähettämään BR2-viestiin. Mikäli RB2-viestissä lähetetty viestilaskurin arvo sekä BR3-viestissä palautettava viestilaskurin arvo eivät täsmää, välimoduuli tunnistaa virheen tiedonsiirrossa.

Taulukko 5. BR3-viesti

Tavu	Parametri	Lisätiedot
0	Vastaus:	
	0x01h	Komento vastaanotettu
	0x02h	Komento suoritettu
	0x03h	Komento vastaanotettu ja suoritettu
1	Viestilaskuri	

BR4-viestissä (taulukko 6) välitetään välimoduulille tieto enimmillään kuudesta WinEEMin digitaalisesta tilasta. Nykyisessä järjestelmässä ainoastaan kahdelle digitaaliselle tilalle on määritelty toiminnallisuutta. BR4-viestin ensimmäisen tavun kolmanteen ja kahdeksanteen bittiin on määritelty käytössä olevan moottorinohjausyksikön versio (EEM3 vai EEM4) ja sen tila (on/off). BR4-viestissä saapuneet tilat välitetään PUMA:lle välimoduulin digitaalilähdöillä (Dout 1 ja Dout 2).

Taulukko 6. BR4-viesti

Tavu	Bitti	Parametri
0		
	Bitti 8	ECU versio(EEM3/EEM4)
	Bitti 7	-
	Bitti 6	-
	Bitti 5	-
	Bitti 4	-
	Bitti 3	ECU power (on/off)

WinEEM lähettää BR5-viestissä (taulukko 7) välimoduulille käskyn TSC1-viestin lähetyksen aloituksesta tai lopetuksesta. Viestissä on lisäksi skaalaustiedot analogisen signaalin käsittelyä varten ja tieto valitusta vääntökäyrästä.

Taulukko 7. BR5-viesti

Tavu	Parametri	Lisätiedot
0	Nopeus-/vääntöpyynnön aktivointi:	
	0x00h	Lopeta viestin lähetys
	0x01h	Aloita viestin lähetys
1	Nopeuspyynnön tyyppi:	
	0x00h	TSC1
	0x02h	EEM3 MF
2	Haluttu vääntökäyrä	
3,4	Vääntöpyynnön skaalaus 0%	
5,6	Vääntöpyynnön skaalaus 100%	

WinEEM lähettää avauksensa yhteydessä BR6-viestissä (taulukko 8) välimoduulille tunnistetiedot WinEEMin ohjelmistoversiosta. Viestiin määritetyt parametrit (tavut 0...5) ovat suoraan ASCII-taulukon muunnokset kirjaimille ’WINEEM’, kaksi viimeistä tavua määrittelee ohjelmistoversion.

Taulukko 8. BR6-viesti

tavu	parametri
0	0x57h
1	0x49h
2	0x4Eh
3	0x45h
4	0x45h
5	0x4Dh
6	0x33h
7	0xFFh

Välimoduuli vastaa WinEEMin lähettämään BR6-viestiin omalla versiollansa BR7 -viestissä (taulukko 9).

Taulukko 9. BR7-viesti

Tavu	Parametri
0	0x42h
1	0x52h
2	Versio, tavu 1
3	Versio, tavu 2
4	Versio, tavu 3
5	Versio, tavu 4

Välimoduulin ja moottorinohjausyksikön kommunikointi (TSC1-viesti)

BR5-viestiin määritetyn ensimmäisen tavun parametrin mukaan välimoduuli voidaan aktivoida lähettämään SAE J1939:n mukaista TSC1 (Torque Speed Control 1) parametriryhmän CAN-viestiä. TSC1-viestillä moottorinohjausyksikölle välitetään standardin mukainen tieto vääntöpyynnöstä. Vääntöpyynnön arvo on analogisen signaalin (Analog In 1), sekä BR5-viestin skaalausarvojen (tavut 3,4,5,6) muodostama arvo väliltä 0–100 %. Liitteen 2 vuokaaviossa on kuvailtu TSC1-viestin lähetystä.

3.2.2 Kättelyt

WinEEM – Välimoduuli

WinEEM-kehitysohjelmiston avauksen yhteydessä ohjelmisto lähettää väylälle viestin WinEEMin versiosta (BR6-viesti). Välimoduulin vastaanottaessa BR6-viestin, se muodostaa omasta versiostaan BR7-viestin ja lähettää sen vastauksena WinEEMille.

Onnistuneen kättelytapahtuman jälkeen kommunikointiyhteys on alustettu ja varsinainen tiedonsiirto voidaan aloittaa.

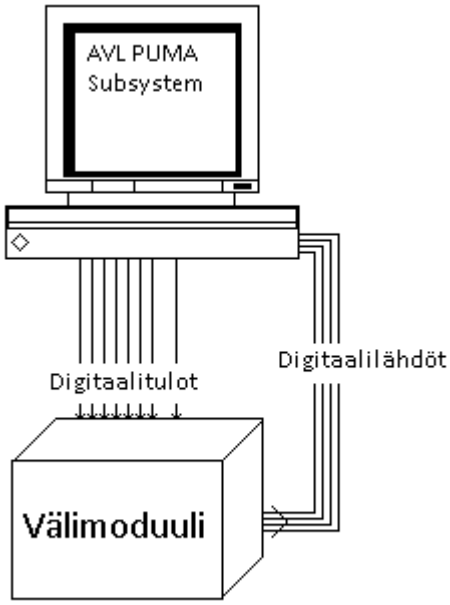
WinEEM - (Välimoduuli) – PUMA

WinEEM vastaa välimoduulin välittämään, BR2-viestissä saapuvaan PUMA-komentoon lähettämällä BR3-kuittausviestin. Mikäli välimoduuli ei saa BR3-viestiä 1000 millisekunnin jälkeen BR2-viestin lähetyksestä, se lähettää BR2-viestin uudelleen. Välimoduuli välittää BR3-kuittausviestin perusteella tiedon suoritetusta komennosta PUMAlle nostamalla yhden digitaalilähdöistä aktiiviseksi.

3.3 Järjestelmän digitaalinen I/O

Välimoduulin digitaalituloilla muodostetaan rinnakkaisväylä PUMAn ja välimoduulin välille (kuva 12). Väylä muodostetaan yhdeksällä johtimella. Jokaisella johtimella voidaan välittää yksi bitti. Yksi biteistä varataan viestin aktivointibitiksi ja lopuilla muodostetaan varsinainen PUMA-komennon muodostava viesti. Toteutus mahdollistaa enimmillään kahdeksan bittisen eli yhden tavun pituisen viestirakenteen. Kahdeksan bitin pituisella viestirakenteella voidaan toteuttaa 256 eri kombinaatiota eli väylän yli voidaan välittää maksimissaan 256 eri PUMA-komentoa.

PUMA-komennot välitetään CAN-väylän yli WinEEMille. WinEEMiin on ennalta määritelty PUMA:n komentoja vastaavat toiminnot (liite 1). Tällä hetkellä komentoja on yhteensä 31 kappaletta, joten nykyisen väylän muodostukseen vaaditaan vähintään kuusi johdinta (5 johdinta tiloille ja yksi aktivointibitille).



Kuva 12. Digitaalinen I/O

Välimoduulin digitaalilähdöillä välitetään PUMA:lle BR4-viestissä vastaanotettu tieto moottorinohjausyksikön versiosta (EEM3 vai EEM4) ja tilasta (on/off).

3.4 Järjestelmän analoginen I/O

Välimoduuli mahdollistaa kolmen analogisen signaalin (Analog In 1 – Analog In 3) käsittelyn. Tämän hetkisessä toteutuksessa ainoastaan Analog In 1 -signaali sisältää toiminnallisuutta. Analog In 1 -signaali on vääntöpyynnön signaali, jonka välimoduuli lukee suoraan potentiometriltä ja luo siitä tarvittaessa CAN-väylälle TSC1-viestiin halutun vääntöpyynnön sekä lähettää synkronista BR1-viestiä WinEEMlle.

3.5 Laiteohjelma

Nykyisen välimoduulin laiteohjelma (ts. siltasovellus) on kirjoitettu C-ohjelmointikielellä. Tässä työssä tullaan hyödyntämään uutta siltasovellusta suunniteltaessa nykyisen välimoduulin lähdekoodia. Laiteohjelman ohjelmointiin perehdytään tarkemmin kappaleessa *5.2 Siltasovelluksen ohjelmointi*.

4 Uusien ratkaisuvaihtoehtojen kartoittaminen

Nykyinen siltasovellus on toteutettu EEM2-moottorinohjausyksikköön. EEM2 -moottorinohjausyksikkö on siltasovelluksen toteuttavana alustana varsin käytännöllinen. Se mahdollistaa kaikkien järjestelmän vaatimien toiminnallisuuksien toteuttamisen AGCO Powerin tuotekehityksen omalla (ts. ennestään tutulla) alustalla. Kyseisen moottorinohjausyksikön tuotanto kuitenkin päättyi vuonna 2005. Jäljellä olevien EEM2-yksiköiden iän ja vähäisen määrän takia siltasovellukselle tarvittiin uusi alusta.

4.1 Vaatimukset

Uudelta välimoduulilta ei vaadita nykyisestä välimoduulista poikkeavia ominaisuuksia. Sen tarkoituksena on toteuttaa samat toiminnallisuudet vastaavilla rajapinnoilla. Alle on listattu järjestelmään kohdistuvat tekniset ja järjestelmätekniset vaatimukset.

Tekniset vaatimukset

- CAN-väylä (HSCAN)
- vähintään 6 digitaalista tuloa
- 1 analoginen tulo
- 3 digitaalista lähtöä
- alusta ohjelmoitavissa

Järjestelmätekniset vaatimukset

- Ratkaisussa käytettyjä komponentteja ja työkaluja on oltava saatavilla vähintään 5 vuotta, mielellään 10 vuotta.
- Ratkaisu on toimintavarma.
- Muutostöiden ja lisäominaisuuksien tekeminen olisi mahdollista, eikä vaadi liikaa resursseja.

Uudelta ratkaisulta vaaditaan vähintään 5 vuoden, mutta mielellään 10 vuoden käyttöikää. Käyttöikää on todennäköisesti yritettävä arvioida itse, sillä mahdollisten erilliskomponenttien suhteelliset toiminta- ja valmistusiät saattavat vaihdella hyvinkin paljon.

Mikäli ratkaisulla olisi mahdollisuus päätyä tuotannossa valmistuvien moottoreiden testausjärjestelmän osaksi, tulisi sen saavuttaa korkea toimintavarmuus. Korkealla toimintavarmuudella tarkoitetaan tässä tapauksessa sitä ettei siltasovellus tai sen alustana toimiva rauta toimi missään tilanteessa väärin.

Tulevat EEM-versiot ja tuotannon eri järjestelmien päivitykset saattavat luoda tarpeen välimoduulin siltasovelluksen muokkaukseen. Tästä syystä siltasovelluksen tulisi olla mahdollisimman vaivattomasti uudelleen ohjelmoitavissa. Muokattavuuden kannalta parhaimmat ratkaisut olisivat joko valmis I/O-moduuliyksikkö tai jokin kehitysalusta, joka olisi ohjelmoitavissa esimerkiksi CAN-väylän yli.

4.2 Vaihtoehtojen rajaus

Vaihtoehtojen kartoitustyö aloitettiin selvittämällä, minkälaisilla rakenteellisilla ratkaisuvaihtoehdoilla uusi välimoduuli olisi toteutettavissa. Vaihtoehdot rajattiin kolmeen mahdolliseen rakenteeseen. Rakenteella tarkoitetaan tässä tapauksessa sitä, minkälaisista lähtökohdista uuden alustan suunnittelu ja rakentaminen aloitettaisiin. Rajauksen jälkeen jokaiselle rakenteelliselle vaihtoehdolle etsittiin siltasovelluksen tarpeita parhaiten palveleva alusta.

Rakenteelliset ratkaisuvaihtoehdot olivat

- täysin itse suunniteltu ja rakennettu piirikortti
- kehitysalusta välimoduulina
- valmis väylämoduuliyksikkö.

4.3 Itsesuunniteltu piirikortti

Oman piirikortin suunnittelu ja rakentaminen olisi uuden välimoduulin toteutusvaihtoehtoehdoista ehdottomasti työläin. Prototyypin suunnittelu tulisi aloittaa käytännössä tyhjältä pöydältä, mikä sinällään antaa jo osviittaa toteutuksen vaatimista ajallisista resursseista. Ratkaisun hyvänä puolena verrattaessa valmiisiin moduuliratkaisuihin olisi moduulin tarpeille tarkoin räätälöity piirikortti sekä laiteohjelma. Se olisi myös kolmesta valitusta ratkaisuvaihtoehdosta todennäköisesti edullisin.

Tuotannossa vaadittava, erittäin korkea toimintavarmuus asettaa omat haasteensa täysin oman raudan ja sovelluksen suunnittelulle. Edellä mainitun ajallisen resurssin lisäksi ratkaisu vaatii huomattavasti enemmän tietotaitoa laitteiston ja siltasovelluksen suunnittelusta ja testaamisesta. Oman piirikortin suunnittelu onkin yleensä suuren volyymin sovelluksille sekä ainoalaatuisille järjestelmille sopivin ratkaisu. Järjestelmän ainoalaatuisuus sinällään tukee tarvetta, mutta siltasovelluksen suhteellisen yksinkertainen C-ohjelma ei kuitenkaan vaadi ainutlaatuista ratkaisua alustakseen.

4.3.1 Suunnittelu ja vaatimukset

Ratkaisussa suunniteltaisiin ja rakennettaisiin piirikortti, jossa CAN-viestejä ja I/O:ta käsittelevänä komponenttina toimisi mikrokontrolleri ja sen laiteohjelma. Oma piirikortti vaatisi mikrokontrollerin lisäksi muita elektroniikan komponentteja, piirilevyn, kotelon sekä liittimet ja johdotuksen.

Kävin läpi eri valmistajien tarjoamia mikrokontrollereita sekä niille tarjottuja kehitysalustoja. Luvussa *3.1 Vaatimukset* esiteltyjen välimoduuliin kohdistuvien teknisten sekä järjestelmäteknisten vaatimusten täyttymisen lisäksi, kiinnitin huomiota seuraaviin, suunnittelutyötä keventäviin tekijöihin:

- mikrokontrolleriin integroitu CAN controller
- mikrokontrollerille löytyisi vastaavan kaltaisia projekteja (esimerkkejä ja kirjastoja)
- laadukas ja riittävän helppokäyttöinen kehitysalusta/ -ympäristö.

Mikrokontrolleriin integroidulla CAN controllerilla säästettäisiin suhteellisen paljon lisäsuunnittelun tarvetta ja lisäksi erilliskomponenttien määrää saataisiin vähennettyä. Mikrokontrollerin olisi hyvä olla mahdollisimman käytetty, jotta eri tietolähteet tarjoaisivat kyseiselle piirille mahdollisimman paljon tietoa sekä mahdollisesti valmiita esimerkkejä. Ratkaisun kannalta oleellista olisi myös mikrokontrollerin valmistajan tarjoamat valmiit ohjelmointikirjastot CAN-väylän ja I/O:n ohjelmointia varten.

4.3.2 AT90CAN128

Parhaaksi ratkaisuksi valikoitui Atmelin valmistama AT90CAN128 -mikrokontrolleri, jossa CAN controller on integroituna mikrokontrolleriin. Kyseiselle mikrokontrollerille löytyi sopiva, kuvassa 13 esitetty kehitysalusta, jolla sovelluksen testaus olisi yksinkertaista. Alustalle löytyi myös jokseenkin vastaavia projekteja, joista voisi olla suhteellisen helppoa lähteä jatkokehittämään sovellusta välimoduulin prototyypille.

Kuva 13. OLIMEX AT90CAN128 -kehitysalusta

Atmel tarjoaa kyseiseen mikrokontrolleriin valmiit kirjastot ja esimerkkejä CAN-väylän viestien käsittelyyn. Kyseisen kehitysalustan valmistajalta (OLIMEX) sai myös ladattua

kehitysalustan piirikaavion, jonka perusteella oman piirikortin pystyisi suunnittelemaan kohtuullisella vaivalla. [10.]

Oman piirikortin edut ovat

- sovellukselle tarkoin räätälöity laitteisto ja laiteohjelma
- edullisuus

Oman piirikortin huonot puolet ovat seuraavat

- vaatii paljon osaamista ja aikaa suunnitteluun ja testaukseen
- piirikorttiin tarvitaan suuri määrä erilliskomponentteja
- saattaa vaatia enemmän resursseja, mitä projektiin on käytettävissä (ts. saavutetaanko esimerkiksi vaadittavaa toimintavarmuutta käytettävissä olevilla resursseilla).

4.4 Kehitysalusta välimoduuliratkaisuna

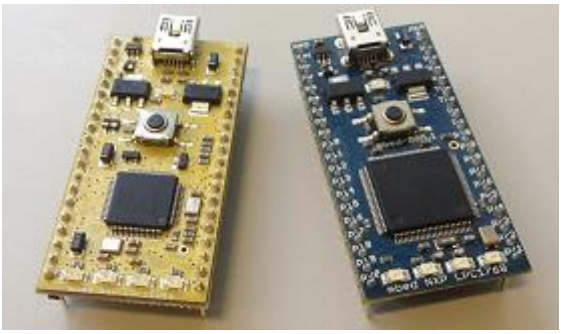
Sulautettujen järjestelmien kehitysalustat ovat elektronisten laitteiden kehityksessä käytettyjä piirikortteja. Niiden käyttäjäkunta koostuu niin teollisuuden tuotekehityksen ammattilaisista kuin myös omia sovelluksia kehittävistä elektroniikan harrastelijoista. Piirikortin perusajatuksena on tarjota laitekehittäjälle helppokäyttöinen kehitysalusta mikrokontrolleripohjaiselle sovellukselle. Perinteisesti kehitysalusta tarjoaa rajapinnan muutamalle teollisuuden automaatioväylälle, I/O-portteja sekä yleensä avoimeen lähdekoodiin pohjautuvan mikrokontrollerin ohjelmointiympäristön.

Mbed-piirikorttisarja vaikutti sopivimmalta CAN-väylälliseltä kehitysalustalta, jolle siltasovellus olisi toteutettavissa. Mbed-piirikortin lisäksi ratkaisu vaatisi yksinkertaisen piirilevyn, kotelon sekä liittimet ja johdotuksen.

Mbed

Mbed Microcontrollers on sulautettujen järjestelmien kehitysalustasarja, joka tällä hetkellä kattaa kaksi kehitysalustana toimivaa piirikorttia *mbed NXP LPC11U24* ja *mbed NXP LPC1768* (kuva 14). Molemmat sarjan kehitysalustoista pohjautuu NXP:n valmistamaan LPC-sarjan mikrokontrollereihin, joiden ytiminä toimivat ARM Cortex-M

-sarjan 32-bitin prosessorit. Ne tarjoavat tehokkaan prosessorin ja pienikokoiseksi kehitysalustaksi laajalti erilaisia väyläratkaisuja. [11.]



Kuva 14. mbed NXP LPC11U24- ja mbed NXP LPC1768 -kehitysalustat

Sarjan kaksi piirikorttia eroavat toisistaan suoritustehon, muistien määrän ja rajapintojen suhteen. Huomattavaa on, että edullisempi LPC11U24 -piirikortti ei tarjoa rajapintaa CAN-väylälle ja on täten poissuljettava sovelluksen toteuttavista ratkaisuvaihtoehdoista. Taulukossa 10 on esitetty mbed NXP LPC1768 -piirikortin ominaisuudet. [11.]

Taulukko 10. mbed NXP LPC1768:n ominaisuudet

Mikrokontrolleri		Oheislaitteet (kpl)
Ydin	ARM Cortex-M3	Ethernet
Taajuus	96Mhz	USBHost
FLASH	512KB	USBDevice
RAM	32KB	SPI (2)
Virrankulutus	60-120m(Vin)	I2C (2)
		CAN (2)
		AnalogIn (6)
		PwmOut (6)
		AnalogOut (1)

Ohjelmointiympäristö

Mbed tarjoaa piirikortin ostajalle ilmaisen, selaimen kautta käytettävän, kevyen ja käyttäjäystävällisen kehitysympäristön. Ohjelmointikielenä toimii C/C++-muunnelma, joka tarjoaa erityisesti piirikortin rajapinnoille valmiit kirjastot. Laitevalmistajan

kotisivuilla on myös suhteellisen aktiivinen foorumi, jossa laudan käyttäjät saavat jakaa projektiensa lähdekoodia ja laatimiansa kirjastoja.

Mbedin selaimen kautta käytettävä ”suljettu” ohjelmointiympäristö saattaa olla siltasovelluksen tulevaisuuden kannalta ongelmallinen. Mikäli kehitysalustan valmistus lopetetaan tai ohjelmointiympäristöä päivitetään taaksepäin yhteensopimattomaan versioon, voi uusien välimoduulien ohjelmointi tai muutosten tekeminen olla mahdotonta tai vähintäänkin työlästä.

Mbedin edut ovat

- helppokäyttöinen kehitysympäristö
- tehokas prosessori ja riittävästi muistia
- CAN-väylä ja riittävästi I/O portteja
- mahdollisuus lisätä lisätoiminnallisuutta helposti

Mbedin huonot puolet ovat seuraavat

- kehitysympäristö on suljettu ja selaimeen sidottu.
- Ei ole tietoa, onko piirikorttia saatavissa vielä 5–10 vuoden päästä.
- Ratkaisu vaatii piirikortin lisäksi vielä ainakin muutaman erilliskomponentin, piirilevyn, kotelon, liittimet ja johdot, jolloin kustannukset saattavat nousta jopa valmiin moduuliyksikön hintaluokkaan.

4.5 Valmis moduuliratkaisu

Uuden välimoduulin toteutusvaihtoehdoista suoraviivaisin ratkaisu olisi suunnitella siltasovellus ohjelmoitavaan, CAN-väylälliseen, I/O-moduuliyksikköön. Kyseisiä moduuliyksiköitä on tarjolla erityisesti automaatioteollisuuden väylätarpeisiin, mutta tietyt valmistajat tarjoavat myös autoteollisuuden väyläratkaisuja tukevia moduuliyksiköitä.

Valmis moduuliyksikkö vapauttaisi prototyypin laitteiston suunnittelu- ja rakentamisvaiheiden ajalliset resurssit täysin siltasovelluksen suunnitteluun ja testaukseen. Ohjelmoitava I/O-moduuliyksikkö antaisi todennäköisesti ratkaisuista korkeimman toimintavarmuuden ja poistaisi oman raudan suunnittelun vaatiman järjestelmäintegroinnin, testauksen ja simuloinnin tarpeen.

AGCO Powerin tuotekehityksessä käytössä olevat PEAK Systemin PCAN-tuotesarjan CAN-väylälliset I/O-moduuliyksiköt olivat luontevin vaihtoehto välimoduulin toteuttavana, valmiina moduuliyksikköratkaisuna. Kyseiselle sarjalle oli olemassa AGCO Powerin aikaisemmissa projekteissa käytetty kehitysalusta, joka olisi käytettävissä prototyypin kehittelyä varten.

4.5.1 PCAN

PEAK System on saksalainen teollisuuden kommunikointiratkaisujen laitteistoon, ohjelmistokehitykseen ja palveluihin erikoistunut yritys. Yrityksen valmistama PCAN-tuotesarja tarjoaa CAN-väylällisiä valmiita I/O-moduuleja, erityisesti auto- ja automaatioteollisuuden tarpeisiin. PCAN-tuoteperheen moduulit ovat kokonaisvaltaisia ratkaisuja, joissa tuotteen käyttäjälle jää ainoastaan halutun sovelluksen toteuttaminen moduuliyksikön rautaan. Käytännössä kaikki sarjan moduulit sisältävät saman emolevyn, mutta moduulien tarjoamat I/O-portit ja rajapinnat ovat tuotekohtaisia.

PCAN-moduulit ovat konfiguroitavissa valmistajan internetsivuilta ladattavalla konfigurointityökalulla. Työkalun tarjoamat toiminnallisuudet ovat suhteellisen rajalliset. Esimerkiksi tämän työn puitteissa tarvittavat funktionaalisuudet eivät ole toteutettavissa konfigurointityökalun avulla. Tästä huolimatta MicroMod-emolevyn mikrokontrolleriin on mahdollista ladata oma laiteohjelma. Oman laiteohjelman laatimista varten Peak Systemin sivuilta on ladattavissa valmiit kirjastot sekä esimerkkiohjelmia.

4.5.2 PCAN-MicroMod Mix 3

MicroMod -sarjan eniten I/O:ta sisältävä moduuliyksikkö on kuvassa 15 esitetty PCAN-MicroMod Mix 3. Moduuli tarjoaa erinomaisen alustan erilaista I/O:ta sekä CAN-väylän vaativalle sovellukselle. Tästä syystä se olisi yksi varteenotettava ratkaisuvaihtoehto siltasovelluksen alustana. Taulukossa 11 on esitetty moduulin ominaisuuksia ja rajapintoja. [12.]

Kuva 15. PCAN-MicroMOD Mix3 -moduuliyksikkö

Taulukko 11. PCAN-MicroMod Mix3 -moduulin ominaisuudet

8 digitaalista tuloa; CMOS-tasot 8 analogista tuloa; 10bit, ref. 5V 4 taajuustuloa
8 digitaalista lähtöä; CMOS-tasot 8 analogista lähtöä 4 taajuus/PWM lähtöä
2 autoteollisuuden liitintä (Tyco) High Speed CAN; ISO 11898-2

Hyvät puolet ovat

- laadukkuus ja toimintavarmuus
- valmis ratkaisu
- tuttuus tuotekehitykselle
- moduulin saatavuus todennäköisesti > 5 vuotta
- olemassa oleva kehitysalusta.

Huonot puolet ovat

- hintavuus
- ei mahdollisuutta kovinkaan suurten lisäominaisuuksien lisäämiseen

4.6 Vaihtoehtojen vertailua

Ratkaisuvaihtoehtojen valinnan jälkeen on mielekästä vertailla vaihtoehtoja keskenään. Jokainen tutkituista vaihtoehdoista täyttää uuden välimoduulin toteuttavalle ratkaisulle asetetut tekniset kriteerit eli kaikilla valituilla vaihtoehdoilla välimoduuli on toteutettavissa. Tässä luvussa käsitellään pääasiassa vaihtoehtojen järjestelmäteknisiä ominaisuuksia. Tärkeimmät järjestelmätekniset vaatimukset olivat käyttöikä, muokattavuus ja toimintavarmuus. Prototyyppiä varten pyritään valitsemaan luonnollisesti paras vaihtoehto jatkokehitykseen. Liitteessä 3 on vertailtu vaihtoehtoja sanallisesti.

Uudelle ratkaisulle asetettiin vähimmäiskäytössäoloiäksi viisi vuotta. Tällä ajalla valitulle ratkaisulle olisi oltava tarjolla kaikki toteutuksessa käytetyt komponentit sekä työkalut, mielellään vastaavanlaisina. Arvioltani kaikki ratkaisuvaihtoehdot saavuttavat viiden vuoden saatavuuden.

Välimoduulin yksi tärkeimmistä vaatimuksista on toimintavarmuus tuotannon testausjärjestelmän osana. Periaatteessa jokaisella ratkaisuvaihtoehdolla olisi mahdollista saavuttaa korkea toimintavarmuus, mutta sekä mbed että itse suunniteltu piirikortti saattavat vaatia tämän työn puitteissa liian paljon aikaa prototyypin suunnitteluun ja testaukseen, jotta riittävän korkea toimintavarmuus saavutettaisiin.

Mahdolliset muutostyöt kohdistuisivat pääasiassa laiteohjelman muokkaamiseen. Kaikilla vaihtoehdoilla laitteen uudelleen ohjelmointi on mahdollista. Kaikkien ratkaisujen uudelleen ohjelmointi ja laiteohjelman jatkokehitys on myös suhteellisen yksinkertaista. Mbedin tapauksessa selaimeen sidottu, suljettu ohjelmistoympäristö saattaa aiheuttaa ongelmia, mikäli esimerkiksi valmiita kirjastoja muutetaan tai päivitetään laitevalmistajan toimesta. Tässä tapauksessa koko laiteohjelma olisi käännettävä uusille kirjastoille.

Kaikki vaihtoehdot ovat kokonaiskustannuksiltaan melko samalla tasolla. Välimoduuliyksiköiden tarve on tällä hetkellä noin 20 kappaletta, joten näin pienelle valmistusvolyymille oman piirikortin suunnittelu ja valmistus voi olla ehkä turhan työläs vaihtoehto.

Prototyyppiä varten valitaan PCAN-MicroMod Mix 3, joka on projektin resurssien ja kohteen kannalta ehdottomasti paras ratkaisu.

5 Prototyypin kehittely ja valmistus

Siltasovelluksen uudeksi alustaksi valittiin Peak Systemin PCAN-MicroMod Mix 3 -moduuliyksikkö. Valinta kohdistui kyseiseen alustaan sen toimintavarmuuden ja yksinkertaisuuden takia. Alustan valinnan jälkeen siirryttiin prototyypin kehittelyyn ja valmistukseen. Kehitteleminen aloitettiin tutustumalla MicroMod Mix 3 -moduuliyksikön sisältämään piirikorttiin ja moduuliyksikölle tarkoitettuun kehitysalustaan. MicroMod -piirikortin mikrokontrolleriin ohjelmoitavan laiteohjelman pohjana voitiin käyttää nykyisen välimoduulin siltasovellusta. Varsinaiseksi tehtäväksi jäi tehdä olemassa olevasta siltasovelluksesta käännös uudelle alustalle ja testata sen toimintaa.

5.1 PCAN-MicroMod

PCAN-MicroMod Mix 3 -moduuli sisältää kuvan 16 mukaisen, irrotettavan piirikortin, josta Peak käyttää nimitystä *motherboard* (suom. *emolevy)*. Piirikortti sisältää Fujitsun valmistaman 16-bittisen, erityisesti ajoneuvosovelluksiin suunnatun MB90F497 -mikrokontrollerin (kuvassa 16 näkyvän piirikortin pohjatasolla).

Kuva 16. PCAN-MicroMod -emolevy

Emolevy on ohjelmoitavissa joko Peakin omalla *PCAN configuration tool* -sovelluksella tai korvaamalla alkuperäinen laiteohjelma omalla laiteohjelmalla. Tämän työn puitteissa *PCAN configuration tool* ei mahdollista kaikkien siltasovelluksessa tarvittavien funktioiden toteuttamista, joten ainoaksi vaihtoehdoksi jäi laatia mikrokontrollerille oma laiteohjelma. Peakin sivuilta sai ladattua kirjastotiedostot, joista löytyi valmiit funktiot moduuliyksikön I/O-porttien ja CAN-väylän käsittelyyn.

Laiteohjelma ohjelmoitiin Fujitsun *Softune Workbench* -kehitysympäristöllä ja C-ohjelmointikielellä. Ohjelma oli helpointa välittää mikrokontrollerille AGCO Powerin

aiemmissa projekteissa käytetyllä, kuvassa 17 esitetyllä PCAN-MicroMod Evalution Kit -kehitysalustalla. Käytännössä kuitenkin MicroMod-emolevyn ohjelmointi tapahtuu RS-232-sarjaliikenneväylän ylitse, joten kehitysalustan käyttö ei ole välttämätöntä.

Kuva 17. MicroMod-emolevyn kehitysalusta.

5.2 Siltasovelluksen ohjelmointi

Uuden siltasovelluksen ohjelmointi aloitettiin tutustumalla nykyisen välimoduulin lähdekoodiin. EEM2-moottorinohjausyksikköön ladattu siltasovellus oli laadittu C-ohjelmointikielellä. Siinä hyödynnettiin ohjausyksikön omia kirjastoja, jotka mahdollistivat mm. säikeistyksen ja semaforien käytön. Siltasovellus on konstruktioltaan kuitenkin suhteellisen yksinkertainen, joten näiden toimintojen käyttö uudessa siltasovelluksessa ei ole välttämätöntä. Ne ovat kuitenkin olleet avaintekijänä nykyisen välimoduulin korkean toimintavarmuuden saavuttamisessa. Alkuperäisen siltasovelluksen päätoiminnot oli toteutettu neljällä rinnakkain ajettavalla säikeellä, joiden tehtävät ovat

- analogisen signaalin käsittely
- rinnakkaisväylän käsittely (digitaaliset tulot)
- TSC1-viestin lähetys
- saapuvien CAN-viestien käsittely.

Siltasovelluksen tehtävien jaottelu kyseisellä periaatteella tuntui järkevältä, joten uutta siltasovellusta lähdettiin laatimaan tämän pohjalta. Ohjelman tehtäviä ei kuitenkaan

ollut mahdollista toteuttaa säikeistyksellä. Sen sijaan kyseiset neljä tehtävää päätettiin luoda hyödyntämällä mikrokontrollerin ajastinfunktiota silmukoiden ajamiseen tietyillä aikasykleillä. Koska funktiot olivat suhteellisen lyhyitä ja ohjelman ei tarvinnut saavuttaa täydellistä reaaliaikaisuutta, ajastimien kellot voitiin sitoa suoraan viestien lähetystaajuuteen. Tällöin esimerkiksi analogisen signaalin käsittelyfunktio ajettiin 50 millisekunnin välein eli BR1-viestin lähetystaajuudella. Liitteessä 4 on kuvattu vuokaavion avulla uuden siltasovelluksen toimintaa.

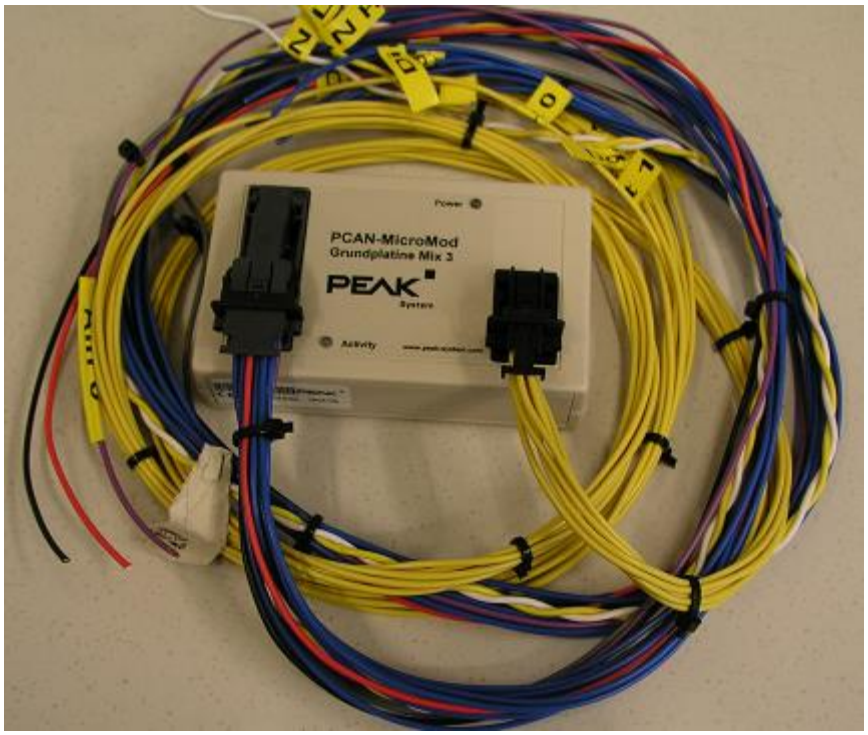
Varsinainen mikrokontrollerin ohjelmointi tapahtui sarjaliikenneväylän yli. Ohjelmointiin tarvittiin USB-RS232-adapteri, jotta Softune Workbenchissä käännetty ’bridge.hex’ -tiedosto saatiin ladattua MicroMod-emolevyn mikrokontrollerin muistiin.

Siltasovelluksen testaus

Uuden siltasovelluksen toimintaa testattiin Softune Workbench -kehitysympäristön testaustyökalulla sekä PCAN-MicroMod Evalution Kit -kehitysalustalla (kuva 18). Testauksissa simuloitiin WinEEM Production toolin ja välimoduulin välistä kommunikointia. Tarkoituksena oli varmistaa kaikkien siltasovelluksen toimintojen oikeanmukainen toiminta.

Kuva 18. Siltasovelluksen testausta PCAN-MicroMod Evalution Kit:llä.

Ohjelman testauksen jälkeen MicroMod emolevy voitiin asentaa PCAN-MicroMod Mix 3 -moduuliyksikköön ja tehdä yksikölle johtosarja. Kuvassa 19 on työn tuloksena valmistunut prototyyppi uudesta välimoduulista.



Kuva 19. Uuden välimoduulin prototyyppi

6 Yhteenveto

Tässä insinöörityössä perehdyttiin AGCO Powerin moottorintestausjärjestelmän välimoduulin päivittämiseen. Työn aiheen motivaationa oli kartoittaa uusia ratkaisuvaihtoehtoja nykyisen siltasovelluksen alustan korvaajaksi. Työssä tarkasteltiin kolmea eri ratkaisuvaihtoehtoa ja arvioitiin niiden soveltuvuutta siltasovelluksen uudeksi alustaksi. Valittujen alustavaihtoehtojen välillä suoritettiin vertailua ja vertailun parhaalle alustalle toteutettiin uusi siltasovellus.

Työssä kehitetty uuden välimoduulin prototyyppi toteutettiin PCAN-MicroMod Mix 3 -moduuliin. Valinta kohdistui kyseiseen alustaan sen yksinkertaisuuden ja toimintavarmuuden takia. Moduulin niin kutsutulle *emolevylle* suunniteltiin C-ohjelmointikielellä siltasovelluksen toteuttava ohjelma, jonka pohjana käytettiin nykyisen, EEM2-moottorinohjausyksikköön toteutetun siltasovelluksen lähdekoodia. Ohjelmaa testattiin kehitysohjelmiston testaustyökaluilla ja MicroMod-sarjalle suunnatulla kehitysalustalla.

Uuden siltasovelluksen pilotointi tuotannossa varsinaisen moottorintestausjärjestelmän osana jäi tämän insinöörityön puitteissa tekemättä. Alusta vaikuttaa kuitenkin lupaavalta vaihtoehdolta nykyisen alustan korvaajaksi.

Lähteet

1 Dietsche, Karl-Heinz. 2010. Dieselmoottorin ohjausjärjestelmät. Helsinki: Autoalan Koulutuskeskus.

2 CAN history. 2013. Verkkodokumentti. CAN in Automation. <http://www.can-cia.org/index.php?id=systemdesign-can-history>. Luettu 6.3.2013.

3 CAN protocol. 2013. Verkkodokumentti. CAN in Automation. <http://www.can-cia.org/index.php?id=systemdesign-can-protocol>. Luettu 6.3.2013.

4 CAN physical layer. 2013. Verkkodokumentti. CAN in Automation. <http://www.can-cia.org/index.php?id=systemdesign-can-physicallayer>. Luettu 9.3.2013.

5 CAN Specification Version 2.0. 1991. Stuttgart: Robert Bosch GmbH.

6 Mahrberg, Teemu. 2013. Induktiivinen kytkeytyminen -kalvosarja. EMC-perusteet -kurssin kurssimateriaali. Metropolia Ammattikorkeakoulu.

7 SAE J1939-71. J1939 Vehicle Application Layer. 2010. SAE International.

8 Voss, Wilfried. 2008. A Comprehensible Guide to 1939. Greenfield: Copperhill Media Corporation.

9 SAE J1939-74. Application-Configurable Messaging. 2010. SAE International.

10 AVR-H128-CAN. 2005. Verkkodokumentti. OLIMEX. <https://www.olimex.com/Products/AVR/Header/AVR-H128-CAN/resources/AVR-H128-CAN.pdf>. Luettu 16.1.2013.

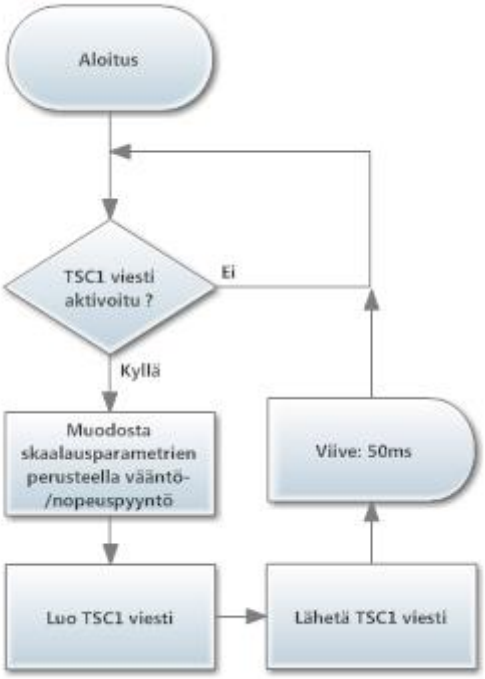
11 Mbed kehitysalusta. 2013. Verkkodokumentti. Mbed. <www.mbed.org>. Luettu 16.1.2013.

12 PCAN-MicroMod Mix 3 -spesifikaatio. 2013. PEAK-System Technik. <http://www.peak-system.com/produktcd/Pdf/English/PCAN-MicroMod-Mix3_UserMan_eng.pdf>. Luettu 7.2.2013.

PUMA-komennot

Komentoarvo	Selitys
0x00	Measurement FULL15
0x01	Verify injection
0x02	Torque curve 1 valinta
0x03	Torque curve 2 valinta
0x04	Torque curve 3 valinta
0x05	Torque curve 4 valinta
0x06	Measurement HIDLE
0x07	Measurement IDLE
0x08	Measurement FULL1
0x09	Measurement FULL12
0x0A	Measurement FULL13
0x0B	Measurement FULL14
0x0C	Measurement FULL3
0x0D	Measurement FULL common
0x0E	Measurement TCURVE
0x0F	Measurement SCURVE
0x11	Torque curve 5 selection
0x12	Close tool
0x13	Adjustment Main Power Correction +
0x14	Adjustment Main Power Correction -
0x15	Adjustment Torque Power Correction +
0x16	Adjustment Torque Power Correction -
0x17	Store Power Correction
0x18	Measurement MODE 1
0x19	Measurement MODE 2
0x1A	Measurement MODE 3
0x1B	Measurement MODE 4
0x1C	Measurement MODE 5
0x1D	Measurement MODE 6
0x1E	Measurement MODE 7
0x1F	Measurement MODE 8

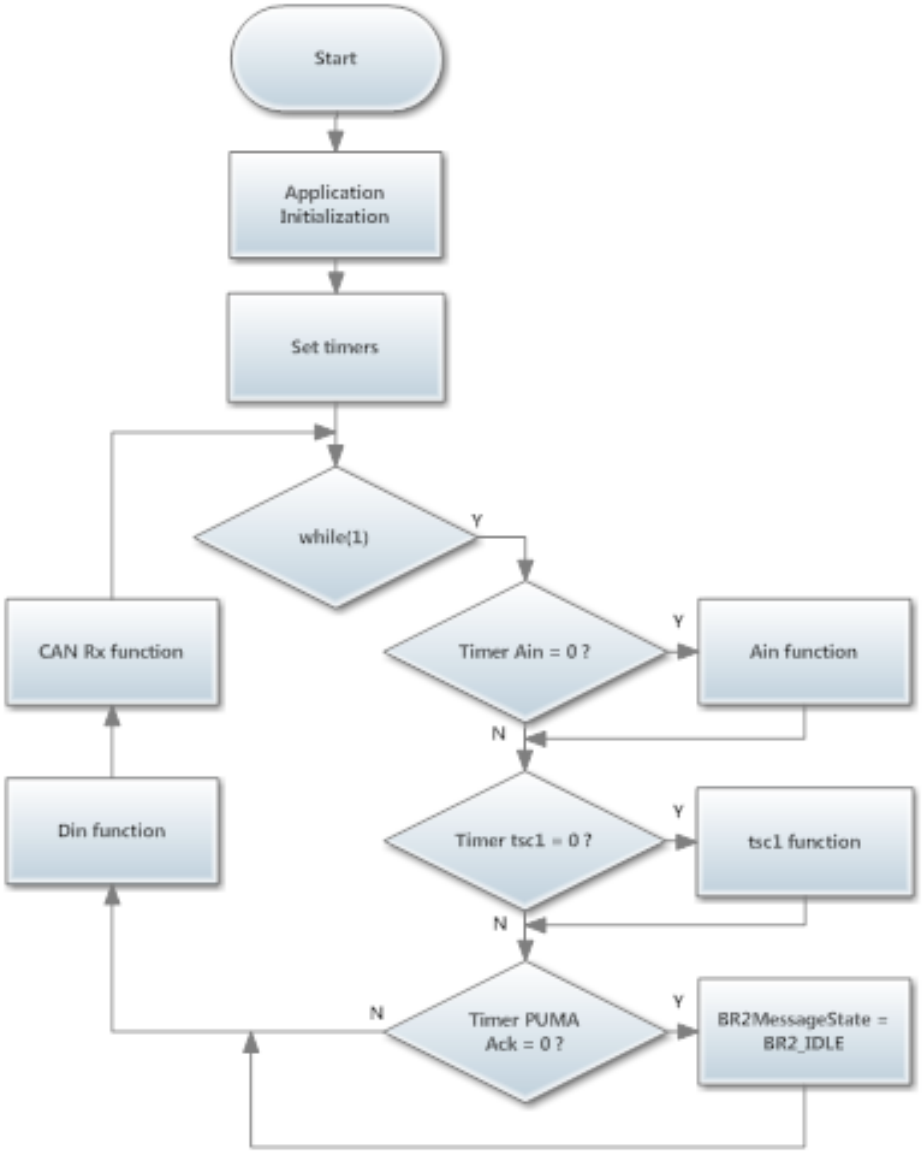
TSC1-viestin lähetys



Vaihtoehtojen vertailua sanallisesti

	Oma piirikortti	**mbed**	**PCAN-MicroMod Mix 3**
Saatavuus tulevaisuudessa	>5 vuotta (oma arvio)	>5 vuotta (oma arvio)	>5 vuotta todennäköisesti > 10 vuotta (oma arvio)
Toimintavarmuus	Riittävän toimintavarmuuden saavuttaminen vaatii erittäin paljon suunnittelua ja testausta	Riittävä toimintavarmuus saavutettavissa, vaatii kuitenkin suhteellisen paljon testausta.	Rauta erittäin toimintavarma. Laiteohjelma vaatii testausta.
Muokattavuus	Laiteohjelma helposti muokattavissa	Laiteohjelma helposti muokattavissa. Miinuksena suljettu ohjelmointiympäristö	Laiteohjelma melko helposti muokattavissa. Ei mahdollista kovinkaan suurten lisäominaisuuksien lisäämistä.
Yksinkertaisuus	Monimutkainen, vaatii paljon suunnittelua ja testausta	Melko yksinkertainen	Yksinkertainen.
Komponentit ja kustannukset	Oma piirilevy, elektroniikan komponentit, kotelo, liittimet, johdot, kehitysalusta. Edullinen.	mbed, oma piirilevy, kotelo, liittimet, johdot, muutama erilliskomponentti. Edullinen.	Valmis paketti. Suhteellisen hintava vaihtoehto

Siltasovelluksen rakenne



Lähdekoodi

```
#include "inc\compiler.h"
#include "inc\mb90495.h"
#include "inc\vectors.h"
#include "inc\can.h"
#include "inc\hardware.h"
#include "inc\uart1.h"
#include "inc\timer.h"
#include "inc\i2c.h"
#include "inc\eeprom.h"
#include "inc\adi.h"
#include "inc\pwm.h"

//EEM version:
#define EEM_STD 0
#define EEM2_MF 1
#define EEM3_MF 2

//BR2 state parameters:
#define BR2_IDLE        0x00
#define BR2_SENT        0x01

//Timers:
#define dinTimer 0
#define ainTimer 1
#define tsc1Timer 2
#define pumaAck 3

//Timeouts:
#define DIN_TIMEOUT     1000 /* 1000 ms */
#define AIN_TIMEOUT     50   /* 50   ms */
#define TSC1_FRE        10   /* 10   ms */
#define PUMA_ACK        100  /* 100  ms */

//CAN ID's:
#define BR1         0x00FF5FFC
#define BR2         0x00FF5EFC
#define BR3         0x0CFF5DF9
#define BR4         0x0CFF5CF9
#define BR5         0x0CFF5BF9
#define BR6         0x0CFF5AF9
#define BR7         0x00FF59FC

#define TSC1        0x0C0000F9
#define MFContReq   0x18FFF004
#define OHECS       0x18FDCBF9


//TSC1 sending state
byte bSendTSC1 = 0;

//EEM version: EEM_STD=0, EEM2_MF=1, EEM3_MF=2
byte bEEMVariant = 0;

//Scaling parameters for TSC1 speed/torque request
byte nTorque = 1;
```

```
byte bOHECSCntr = 40;
//Bridge - ECU speed/torque sending state. PGN0 /// speed/torque re-
quest
byte bSendNow = 1;

//Scaling for Analog in
unsigned short unSpeedRequest = 800;
ulong ulSpeedRequestZero = 800;
ulong ulSpeedRequestHundred = 2500;

//PUMA message counters(starts from 0):
byte currentMsg=0xFF;
byte pumaLatestMsg;

//BR2 messagestate state
byte BR2MessageState = BR2_IDLE;

//Speed/Torque messages
char tsc1[8];
char mfContReqMessage[8];
char oHECSMessage[8];


/*------------------------------------------------------------ *
 * FUNCTION NAME : Timer_2000Hz
 * DESCRIPTION   : Initializes AD + LED
------------------------------------------------------------ */

void Timer_2000Hz(){
    static byte n;

    n++;
    if(n&1) // nur jedes 2. mal aufrufen
    {
        LED_1000Hz();
        AD_1000Hz();
    }
}
/*------------------------------------------------------------ *
 * FUNCTION NAME : APPLInit
 * DESCRIPTION   : Initializes all the components
------------------------------------------------------------ */

void APPInit(void){

    InitIrqLevels();
    Hardware_Init();
    Timer_Init();
    CAN_Init(CAN_BAUD_250K);
    CAN_RegisterMsg(0);

    CAN_TxCallback = 0; // assignment of Callback-Function

    EnableExtendedID(1);
    CAN_Online();
    AD_Init();
    AD_1000Hz();
    __EI();
}
```

```
/*------------------------------------------------------------ *
 * FUNCTION NAME : Ain
 * DESCRIPTION   : The AIN reader function reads AIN1 signal
                   at rate of AIN_TIMEOUT and sends info to WinEEM.
------------------------------------------------------------ */

void Ain(void){

    word value;

    Timer_Set(ainTimer,AIN_TIMEOUT);
    value = AD_Read(0);

    unSpeedRequest = ulSpeedRequestZero + ((ulSpeedRequestHundred -
ulSpeedRequestZero)*(value/1023.0));

    CAN_BUFF_ID=BR1;
    CAN_BUFF_LEN=8;
    CAN_BUFF_DATA[0]=(byte)value;
    CAN_BUFF_DATA[1]=(byte)(value>>8);
    CAN_BUFF_DATA[2]=0;
    CAN_BUFF_DATA[3]=0;
    CAN_BUFF_DATA[4]=0;
    CAN_BUFF_DATA[5]=0;
    CAN_BUFF_DATA[6]=0xFF;
    CAN_BUFF_DATA[7]=0xFF;
    CAN_BUFF_FORMAT = CAN_EXT;  // extended 29 bit

    CAN_Write();

    switch (bEEMVariant) {
        case EEM2_MF:
            mfContReqMessage[1] =(unSpeedRequest >> 8) & 0xFF;
            mfContReqMessage[0] =unSpeedRequest & 0xFF;
            break;
        case EEM3_MF:
        case EEM_STD:
        default:
            tsc1[2] =((unSpeedRequest*8) >> 8) & 0xFF;
            tsc1[1] =(unSpeedRequest*8) & 0xFF;
            break;
        }

    WATCHDOG;
}
```

```
/*------------------------------------------------------------ *
 * FUNCTION NAME : Din
 * DESCRIPTION   : The DIN reader function polles DTR signal.
                   When DTR is UP, sends message to WinEEM.
------------------------------------------------------------ */

void Din(void){

    byte bData;

    /* Triggers to rising edge. Wait input to go down. */
    //while(Get_DIN(7)){}

    if(BR2MessageState == BR2_IDLE){

        Set_DOUT(0,0); //Set dout0 to inactive (no pending BR2 messag-
es). (Equal to PumaSetBusy function)

         /* If signal UP */
        if(Get_DIN(7)){
            currentMsg++;
            pumaLatestMsg=currentMsg;
            Set_DOUT(1,1);

            /* Read databits from Digital Inputs */
                bData=0;
                if(Get_DIN(0)){
                    bData  = 0x01;
                }
                if(Get_DIN(1)){
                    bData |= 0x02;
                }
                if(Get_DIN(2)){
                    bData |= 0x04;
                }
                if(Get_DIN(3)){
                    bData |= 0x08;
                }
                if(Get_DIN(4)){
                    bData |= 0x10;
                }
                if(Get_DIN(5)){
                    bData |= 0x20;
                }
                if(Get_DIN(6)){
                    bData |= 0x40;
                }

                Set_DOUT(0,1);  //Set dout1 to "active" (pending BR2
message) and set pumaAck timer(wait time for PUMA). (Equal
to"PumaSetReady" -function)
                Timer_Set(pumaAck,PUMA_ACK);
```

```
                /*Form BR2 msg */
                CAN_BUFF_ID=BR2;
                CAN_BUFF_LEN=8;
                CAN_BUFF_DATA[0]=bData;
                CAN_BUFF_DATA[1]=currentMsg;
                CAN_BUFF_DATA[2]=0;
                CAN_BUFF_DATA[3]=0;
                CAN_BUFF_DATA[4]=0;
                CAN_BUFF_DATA[5]=0;
                CAN_BUFF_DATA[6]=0;
                CAN_BUFF_DATA[7]=0;
                CAN_BUFF_FORMAT = CAN_EXT;  // extended 29 bit

                CAN_Write();
                //Set dinTimer to 1000ms(response time for WinEEM) and
set BR2MessageState to BR2_SENT;
                Timer_Set(dinTimer,DIN_TIMEOUT);
                BR2MessageState = BR2_SENT;

            //while(Get_DIN(7)){}
        }
    }
}


/*------------------------------------------------------------ *
 * FUNCTION NAME : Tsc1
 * DESCRIPTION   : The function writes TSC1 message to the CAN
                   bus rate at of TSC1_FRE.
------------------------------------------------------------ */

void Tsc1(void){
    //Set timer tsc1Timer to TSC1_FRE
    Timer_Set(tsc1Timer,TSC1_FRE);
    //if bSendTSC1 = 1 (set in BR5-message) start streaming TSC1-
message
    if (bSendTSC1) {
        //ECU version:
        switch (bEEMVariant) {

        case EEM2_MF:
            CAN_BUFF_ID=MFContReq;
            CAN_BUFF_LEN=8;
            CAN_BUFF_DATA[0]=mfContReqMessage[0];
            CAN_BUFF_DATA[1]=mfContReqMessage[1];
            CAN_BUFF_DATA[2]=0xFF;
            CAN_BUFF_DATA[3]=0xFF;
            CAN_BUFF_DATA[4]=0x00;
            CAN_BUFF_DATA[5]=0xFF;
            CAN_BUFF_DATA[6]=0xFF;
            CAN_BUFF_DATA[7]=0xFF;
            CAN_BUFF_FORMAT = CAN_EXT;  // extended 29 bit
            CAN_Write();
            break;
```

```
        case EEM3_MF:
            if (bOHECSCntr >= 40) {
                CAN_BUFF_ID=OHECS;
                CAN_BUFF_LEN=8;
                CAN_BUFF_DATA[0]=0xFF;
                CAN_BUFF_DATA[1]=oHECSMessage[1];
                CAN_BUFF_DATA[2]=0xFF;
                CAN_BUFF_DATA[3]=0xFF;
                CAN_BUFF_DATA[4]=0xFF;
                CAN_BUFF_DATA[5]=0xFF;
                CAN_BUFF_DATA[6]=0xFF;
                CAN_BUFF_DATA[7]=0xFF;
                CAN_BUFF_FORMAT = CAN_EXT;  // extended 29 bit
                CAN_Write();
            }

            if (bSendNow) {
                CAN_BUFF_ID=TSC1;
                CAN_BUFF_LEN=8;
                CAN_BUFF_DATA[0]=0x01;
                CAN_BUFF_DATA[1]=tsc1[1];
                CAN_BUFF_DATA[2]=tsc1[2];
                CAN_BUFF_DATA[3]=0xFF;
                CAN_BUFF_DATA[4]=0xFF;
                CAN_BUFF_DATA[5]=0xFF;
                CAN_BUFF_DATA[6]=0xFF;
                CAN_BUFF_DATA[7]=0xFF;
                CAN_BUFF_FORMAT = CAN_EXT;  // extended 29 bit
                CAN_Write();
            }
            bSendNow = !bSendNow;
            break;

        case EEM_STD:
        default:
            if (bSendNow) {
                CAN_BUFF_ID=TSC1;
                CAN_BUFF_LEN=8;
                CAN_BUFF_DATA[0]=0x01;
                CAN_BUFF_DATA[1]=tsc1[1];
                CAN_BUFF_DATA[2]=tsc1[2];
                CAN_BUFF_DATA[3]=0xFF;
                CAN_BUFF_DATA[4]=0x3F;
                CAN_BUFF_DATA[5]=0x03;
                CAN_BUFF_DATA[6]=0x01;
                CAN_BUFF_DATA[7]=0xFF;
                CAN_BUFF_FORMAT = CAN_EXT;  // extended 29 bit
                CAN_Write();
            }
            bSendNow = !bSendNow;
            break;
    }
}
```

```
    //Parameter for OHECS message
    if (bOHECSCntr >= 40) {
        bOHECSCntr = 0;
    }
    else {
        bOHECSCntr++;
    }
}

/*------------------------------------------------------------ *
 * FUNCTION NAME : CANRx
 * DESCRIPTION   : The CAN receiver function handles received CAN mes-
sages.
------------------------------------------------------------ */

void CANRx(void){
    word i;
    ulong ulMsgID;

    if(CAN_Read()==CAN_ERR_OK){

        ulMsgID = CAN_BUFF_ID;
        switch (ulMsgID) {

        case BR3:
            if((CAN_BUFF_DATA[0]&0x01)==0x01){
                if(CAN_BUFF_DATA[1]==currentMsg){
                    Timer_Set(dinTimer,0);
                    BR2MessageState = BR2_IDLE;
                }
            }
            if(CAN_BUFF_DATA[0]==0x02){
                if(currentMsg == pumaLatestMsg){
                    Set_DOUT(1,0);
                }
            }
        break;

        case BR4:
            WATCHDOG;
           //bitmask 1111 1100 (first 2 douts for BR2-BR3)
           Set_DOUTByteMask(0xFC,CAN_BUFF_DATA[0]);
            WATCHDOG;
        break;

        case BR5:
            if( (CAN_BUFF_DATA[0] == 0x00) ){
                bSendTSC1 = 0;
            }
            else{ //start TSC1 message stream (bSendTSC1 = 1)

                nTorque = CAN_BUFF_DATA[2];

                        ulSpeedRequestZero = CAN_BUFF_DATA[3] +
                                        (CAN_BUFF_DATA[4] << 8);
                        ulSpeedRequestHundred = CAN_BUFF_DATA[5] +
                                        (CAN_BUFF_DATA[6] << 8);
```

```
                    bEEMVariant = CAN_BUFF_DATA[1];

                    switch (bEEMVariant) {
                    case EEM2_MF:
                        mfContReqMessage[4] = nTorque-1;
                        break;
                    case EEM3_MF:
                        oHECSMessage[1] = nTorque-1;
                        for (i=3; i<8; i++) {
                            tsc1[i] = 0xFF;
                        }
                        break;

                    case EEM_STD:
                        tsc1[6] = nTorque;
                        break;
                    }
                    bSendTSC1 = 1;
                    }
                break;

                case BR6:
                    CAN_BUFF_ID=BR7;
                    CAN_BUFF_LEN=8;
                    CAN_BUFF_DATA[0]=0x42;
                    CAN_BUFF_DATA[1]=0x52;
                    CAN_BUFF_DATA[2]=0x00;
                    CAN_BUFF_DATA[3]=0x4A;
                    CAN_BUFF_DATA[4]=0x55;
                    CAN_BUFF_DATA[5]=0x4B;
                    CAN_BUFF_DATA[6]=0x4B;
                    CAN_BUFF_DATA[7]=0x41;
                    CAN_BUFF_FORMAT = CAN_EXT;  // extended 29 bit
                    CAN_Write();
                break;
                }
        }
}
```

```
void main(){

    APPInit();
    //Set timers
    Timer_Set(ainTimer,AIN_TIMEOUT);
    Timer_Set(dinTimer, DIN_TIMEOUT);
    Timer_Set(tsc1Timer,TSC1_FRE);

    while(1){
        WATCHDOG;
        //run function Ain if ainTimer = 0
        if(Timer(ainTimer)==0){Ain();}
        //run function Tsc1 if tsc1Timer = 0
        if(Timer(tsc1Timer)==0){Tsc1();}

        //if  WinEEM hasn't responded in 1000ms. BR2MessageState will
        be set to idle so BR2 message can be send again
        if(Timer(dinTimer) == 0 && BR2MessageState == BR2_SENT)
        {BR2MessageState = BR2_IDLE;}
        //run Din function
        Din();
        //set Digital Out 0 to inactive if pumaAck = 0
        if(Timer(pumaAck)==0){Set_DOUT(0,0);}
        //run CANRx function
        CANRx();
    }
}
```